Collaborative Large-scale Integrating Project

**OPENCOSS**

Open Platform for EvolutioNary Certification Of Safety-critical Systems

Implementation of the process-specific service infrastructure

D7.5

<table>
<thead>
<tr>
<th>Work Package:</th>
<th>WP7: Transparent Certification and Compliance-aware Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination level:</td>
<td>Public</td>
</tr>
<tr>
<td>Status:</td>
<td>Final</td>
</tr>
<tr>
<td>Date:</td>
<td>March 4th, 2015</td>
</tr>
<tr>
<td>Responsible partner:</td>
<td>Janusz Studzizba (Parasoft S.A.)</td>
</tr>
<tr>
<td>Contact information:</td>
<td><a href="mailto:januszst@parasoft.com">januszst@parasoft.com</a></td>
</tr>
</tbody>
</table>

**PROPRIETARY RIGHTS STATEMENT**

This document contains information that is proprietary to the OPENCOSS Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the OPENCOSS consortium.
Contributors

<table>
<thead>
<tr>
<th>Names</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janusz Studziżba</td>
<td>Parasoft S.A.</td>
</tr>
<tr>
<td>Yaping Luo, Alexander Mois Aroyo</td>
<td>Eindhoven University of Technology</td>
</tr>
<tr>
<td>Eric Boccanfuso, Eric Andrianarison</td>
<td>Atego France</td>
</tr>
<tr>
<td>Angel López, Huáscar Espinoza</td>
<td>Tecnalia</td>
</tr>
<tr>
<td>Jérôme Lambourg</td>
<td>Adacore</td>
</tr>
<tr>
<td>Jose Luis de la Vara, Rajwinder Kaur Panesar-Walawege, Carlo Ieva</td>
<td>Simula Research Laboratory</td>
</tr>
</tbody>
</table>

Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>2014-08-01</td>
<td>Document creation, initial ToC, and initial content</td>
</tr>
<tr>
<td>V0.2</td>
<td>2014-08-16</td>
<td>Metrics chapters added</td>
</tr>
<tr>
<td>V0.3</td>
<td>2014-09-11</td>
<td>Implemented tools screenshots added</td>
</tr>
<tr>
<td>V0.4</td>
<td>2014-09-12</td>
<td>Integration of Atego Process Director</td>
</tr>
<tr>
<td>V0.5</td>
<td>2014-10-02</td>
<td>Update of chapter Integration of Atego Process Director</td>
</tr>
<tr>
<td>V0.6</td>
<td>2014-10-06</td>
<td>Implementation overview of Atego Process Director Integration</td>
</tr>
<tr>
<td>V0.7</td>
<td>2014-10-17</td>
<td>Update of Metrics Chapter</td>
</tr>
<tr>
<td>V0.8</td>
<td>2014-11-26</td>
<td>Document reviewed</td>
</tr>
<tr>
<td>V0.9</td>
<td>2014-12-09</td>
<td>Document updated</td>
</tr>
<tr>
<td>V1.0</td>
<td>2015-03-04</td>
<td>Final review</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

Abbreviations ............................................................................................................................ 7
Executive Summary.................................................................................................................... 8
1 Functionality implemented ............................................................................................. 9
2 Implementation architecture and source code description ............................................. 11
  2.1 Implementation modules architecture...................................................................... 11
  2.2 Source code description ............................................................................................. 12
    2.2.1 Server source code................................................................................................. 12
    2.2.2 Client source code ................................................................................................. 14
  2.3 Technologies used for OPENCOSS platform implementation ................................... 15
  2.4 Data storage implementation .................................................................................... 15
3 Integration with external process tools ......................................................................... 17
  3.1 Final approach to process API implementation ......................................................... 17
  3.2 Process definition and execution ............................................................................... 17
  3.3 Integration with Atego Process Director ................................................................... 18
    3.3.1 Overview ................................................................................................................ 19
    3.3.2 Functionality point of view .................................................................................... 20
    3.3.3 Technical overview................................................................................................. 28
    3.3.4 Architecture of Atego Process Tool implementation .................................... 28
4 Project compliance estimation with Compliance report ................................................. 30
  4.1 Goal of the report ...................................................................................................... 30
  4.2 Viewing compliance data on the report .................................................................... 31
  4.3 Adding evidence and compliance data ...................................................................... 34
  4.4 Generation of summary textual report ..................................................................... 36
5 Metrics implementation ................................................................................................ 38
  5.1 Metrics selected for implementation ........................................................................ 38
  5.2 Metrics implementation details.................................................................................. 40
    5.2.1 Metrics derived from Mapping Metamodel .................................................. 40
    5.2.2 Metrics derived from AssuranceAsset Metamodel ....................................... 41
    5.2.3 Metrics derived from Baseline Metamodel ................................................... 43
    5.2.4 Metrics derived from Reframework and Baseline Metamodel .................. 45
5.2.5 Metrics derived from Process Metamodel ................................................................. 45
5.2.6 Metrics derived from Argumentation Metamodel .................................................. 47
5.2.7 Other Metrics Related to Safety Assurance and Certification Process ........ 49
5.3 Metrics Estimation Report .......................................................................................... 52
5.4 Equivalence Map Report .......................................................................................... 54
5.5 Source code description ......................................................................................... 55
6 OPENCOSS platform tool user manual ................................................................... 57
7 OPENCOSS platform tool developer manual ......................................................... 58
8 Conclusion .................................................................................................................. 59
TABLE OF FIGURES

Figure 1: Compliance Estimation web report GUI ................................................................. 10
Figure 2: OPENOSS platform tools - technologies and modules communication .................. 11
Figure 3: OPENOSS platform server source code packages .................................................... 12
Figure 4: SVN structure plugin .............................................................................................. 14
Figure 5: Data storage used by OPENOSS platform modules .................................................. 16
Figure 6: Process definition: data flow .................................................................................. 18
Figure 7: Process execution .................................................................................................... 18
Figure 8: Architecture of Atego Process Tool (APT) .............................................................. 19
Figure 9: Main features in terms of process scope ................................................................. 20
Figure 10: Main features of APT .......................................................................................... 21
Figure 11: Login page* of APT (*Current Version, may change) ........................................... 22
Figure 12: Server configuration of APT (*Current Version, may change) ......................... 22
Figure 13: Current servers setting for APT (*Current Version, may change) ....................... 23
Figure 14: Login to APT (*Current Version, may change) .................................................... 23
Figure 15: Features of APT (*Current Version, may change) ................................................. 24
Figure 16: Process & Project Management page* of APT (*Current Version, may change) .... 25
Figure 17: “Process Reference model” page of APT .............................................................. 25
Figure 18: “Import Process Definition” page* of APT (*Current Version, may change) ...... 26
Figure 19: Import Process Definition Checks page* of APT (*Current Version, may change) .. 26
Figure 20: “Process Baseline model” page of APT ................................................................. 27
Figure 21: “Import Project Definition” page* of APT (*Current Version, may change) ........ 27
Figure 22: Context and Architecture of APT implementation (*Current Version, may change) 29
Figure 23: Menu item directing to “Compliance report” ............................................................ 30
Figure 24: Baseline Frameworks combo box for the specific project ..................................... 31
Figure 25: 4 panels of "Compliance report" ......................................................................... 31
Figure 26: Description of the selected baseline element presented at the bottom panel ........... 32
Figure 27: Details of Justification and mapped evidence ......................................................... 33
Figure 28: Compliance evidence of the specific baseline asset ................................................ 34
Figure 29: Specific evidence details description presented at the bottom ................................. 34
Figure 30: A window allowing to assign and describe evidence to the given baseline item ...... 35
Figure 31: Unassign button allowing to disassociate evidence from the given baseline item ..........36
Figure 32: “Export to MS Word” button which generates textual overall detailed report of Project Compliance to the safety standard...........................................................................................................36
Figure 33 First page of the generated textual report .................................................................................................................................37
**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>CCL</td>
<td>Common Certification Language</td>
</tr>
<tr>
<td>DAO</td>
<td>Data Access Object</td>
</tr>
<tr>
<td>DX.Y</td>
<td>OPENCOSS deliverable X.Y</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modelling Framework</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>SACM</td>
<td>Structured Assurance Case Metamodel</td>
</tr>
<tr>
<td>SVN</td>
<td>Subversion</td>
</tr>
<tr>
<td>TX.Y</td>
<td>OPENCOSS task X.Y</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification and Validation</td>
</tr>
<tr>
<td>WP</td>
<td>OPENCOSS Work Package</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>EEF</td>
<td>Extended Editing Framework</td>
</tr>
<tr>
<td>CDO</td>
<td>Connected Data Objects</td>
</tr>
</tbody>
</table>
Executive Summary

This document presents the fifth deliverable of WP7, which comprises work done in scope of T7.4 for the implementation of a management infrastructure for transparent certification and compliance-aware project process. The aim of this document is to demonstrate the results of this implementation in the OPENCOSS tool platform.

The OPENCOSS project has agreed to divide software development activities into 3 phases, each resulting in a functional and ready-to-use tool prototype. As far as WP7 functionality implementation is concerned, the following work has been done in each phase:

- Phase 1 (resulting in prototype 1)
  - Implementation of Eclipse editors for CCL process metamodel
  - Initial design of compliance estimation reports
- Phase 2 (resulting in prototype 2)
  - Design and initial implementation of API for integration with external process tools
  - Second iteration of design of compliance estimation reports
- Phase 3 (resulting in prototype 3)
  - Implementation of compliance estimation reports
  - Implementation of integration with external process tools
  - Implementation of metrics functionality

The functionality implemented as a result of work performed in WP7 is broader than process management itself. The developed platform tools support the following 3 functional areas:

- Process management of safety project and its certification - including integration with external process tools
- Functionality for safety standard compliance estimation
- Metrics of project and process

This document presents in detail the pieces of functionality implemented in the OPENCOSS platform tools for the 3 areas above, their software architecture, the technology used, and source code references.

Other important parts of D7.5 document are:

- A reference to the User Manual, describing installation and usage of the OPENCOSS tool platform.
- A reference to the Developer Manual, describing a configuration of the software development environment for the OPENCOSS tool platform.
1 Functionality implemented

The following pieces of functionality have been implemented as the result of software development performed in WP7:

- General functionality - transversal to WP4, WP5, WP6, and WP7
  - Client-server infrastructure
  - Central data storage used by both server and clients
- WP7-specific functionality
  - Editors for process management
  - Integration with external process tools
  - Compliance estimation reports
  - Metrics of project and process assets

Software implementation of the above functionality is described in detail in the following chapters.

The main groups of low level requirements, as presented in Chapter 4 of D7.2 document, are shown below. The implementation of each group has been generally described with references to their detailed characterization in the subsequent parts of this document.

**Requirement group “4.1 Mapping of process models”** describes functionality of process definition and its mapping to a specific reference framework. It has been implemented in the Eclipse CCL client editors of the OPENCOSS tool platform.

**Requirement group “4.2 Process execution”** defines functionality for execution and monitoring of a process. In the OPENCOSS platform implementation the execution is possible in two ways:
- Using external process execution tools integrated with the OPENCOSS tool platform, which has been described in Integration with external process tools.
- To some extent, it can be done using OPENCOSS Eclipse client by editing CCL models data.

**Requirement group “4.3 Estimation of compliance”** represents functionality which assists and helps project participants (including safety managers and external assessors) in estimating the current compliance of their project to the selected safety standard. It has been implemented as compliance estimation reports functionality and has been described in Compliance Estimation functionality chapter.

**Requirement groups “4.4 Safety Assurance and Certification Process Metrics”, “4.5 Safe Product Metrics” and “4.6 Safe Process Metrics”** specify metrics for different kinds of assets in the safety project. This functionality has been implemented in the metrics module described in Metrics implementation chapter.
Figure 1: Compliance Estimation web report GUI
2 Implementation architecture and source code description

2.1 Implementation modules architecture

The implemented OPENCOSS platform consists of the following main modules:

- OPENCOSS clients - facilitating data edition
- OPENCOSS web server - facilitating data reporting
- OPENCOSS data storage - used by both clients and the server

This architecture, together with the communication between its main implementation modules, has been described in detail in D6.6 document in “2.1 Client-server architecture with central data storage” chapter. As a brief reminder, the below diagram shows the modules architecture and their communication.

Figure 2: OPENCOSS platform tools - technologies and modules communication
2.2 Source code description

This chapter presents the structure of source code packages of both the server and clients of the OPENCOSS platform implementation. The source code has been developed in the scope of T7.4 implementation effort. Both the client and the server have been implemented using Java language.

Various technology libraries have been used. They are described in Technologies used for implementing the OPENCOSS platform. The code has been committed to the OPENCOSS SVN source control repository.

It is hosted at the following URL: https://svn.win.tue.nl/repos/opencoss-code

2.2.1 Server source code

The source code that implements the OPENCOSS platform server has been committed to the following location: https://svn.win.tue.nl/repos/opencoss-code/trunk/common.

Most of the source code packages presented in the below picture are responsible for common architecture, data storage, and communication. They have been described in detail in D6.6 document in “2.2 Source code description” chapter.

However, the 2 packages selected on the below picture (“org.opencoss.webapp.reports” and “org.opencoss.webapp.processmgr”) constitute the implementation of process- and reports-related functionality on the server side.

Below we provide a general description of the packages responsible for WP7 functionality.

org.opencoss.webapp.reports - This package contains the implementation of OPENCOSS platform server web pages. The functionality has been developed using Vaadin, which is a Java framework for building modern web applications (http://www.vaadin.com). This package general structure has been described in detail in D6.6 document.

![Figure 3: OPENCOSS platform server source code packages](image-url)
From WP7 functionality perspective, the `org.opencoss.webapp.reports` package contains the implementation of Compliance Estimation reports. Its source code is described below.

The report main class is `org.opencoss.webapp.reports.view.ComplianceEstimationReport`. Its source code has been developed following multiple design patterns in order to facilitate robust and state-of-the-art implementation. As all OPENCOSS web reports, it has been implemented according to **Model-View-Controller (MVC) design pattern**. `ComplianceEstimationReport` plays the role of a controller and is a hub for both data presentation and its retrieval from data storage. The `getMainComponent()` method assembles a view - GUI of the report - which consists of four panels. The details about compliance estimation GUI and its functionality have been described in **Compliance Estimation functionality** chapter.

The GUI left-hand-side table presenting specific baseline framework items has been implemented in `ProjectBaselineComplianceStatusTable` class. This table has some functionality in common with the similar table on Gap Analysis report - thus a common functionality for these two tables has been extracted to a parent abstract `AbstractProjectBaselineComplianceTable` class from which both tables inherit.

The model component of the MVC pattern is materialized by **Data-Access-Objects (DAO) design pattern** classes implemented in `org.opencoss.webapp.reports.dao` package. These classes prepare SQL queries and use CDO connectivity functions to retrieve data from OPENCOSS storage.

Another feature of the report implementation worth to mention is its usage of **Observer design pattern**. It is used in several places in the compliance report implementation which concern user interactions with the report. The pattern is used in the following places:

- **When a project is changed in the project combo box on the top of the report** (implemented in `ProjectAndMenuPanel` class), all registered `ProjectChangeListener` objects are notified. `ComplianceEstimationReport` extends `AbstractReport` which implements `ProjectChangeListener` and thus gets notification about the project change. Upon this event, the new project baseline frameworks are read from the storage, and the first baseline framework is selected in baseline frameworks combo box at the top of the report.

- **Baseline frameworks combo box has its own listeners**, and upon its selection (which includes the selection of a new project described in the bullet above), its listeners are notified. The newly selected `baselineFrameworkId` is retrieved and injected into baseline compliance table presented in the left-hand side. The table, implemented in `ProjectBaselineComplianceStatusTable`, gets notified about the newly selected baseline framework, its data is read from the storage and presented in the table.

- **When user selects the specific `ProjectBaselineComplianceStatusTable` row**, its registered Observers are notified. The listener registration is done by the following code in this class: 
  ```java
  projectBaselineComplianceStatusTable.addValueChangeListener(new ValueChangeListener() { … });
  ```
  There are as much as 3 listeners of this event - because 3 panels need to be refreshed upon new baseline item selection:
  - `BaselineItemDetailsPanel` class, which implements the description panel presenting the textual properties of the selected baseline item - placed at the bottom of the report GUI.
Implementation of the evidence management service infrastructure

- **ComplianceDetailsStatusPanel** class - presenting the details of justification, evidence and activity pieces mapped with compliance mapping to the selected baseline table item - presented in the right panel.
- **DragAndDropComplianceEvidencePanel** class, which implements resource file upload panel - this panel needs to be refreshed depending whether user clicks a BaseArtefact or BaseActivity in the baseline items table.

**org.opencoss.webapp.processmgr** - The package contains the implementation of integration with external process tools. Integration with Atego Process Director has been implemented in particular. It provides a REST API that offers services of process data management in OPENCOSS database. The details of the implementation have been described in [Integration with external process tools](#).

### 2.2.2 Client source code

The source code implementing the OPENCOSS platform clients has been committed to the following location: [https://svn.win.tue.nl/repos/opencoss-code/trunk/prototype/plugins](https://svn.win.tue.nl/repos/opencoss-code/trunk/prototype/plugins). The following Java packages that are related with the process implementation:

- **org.opencoss.pam.procspec** - In this plugin, the process execution metamodel is defined and stored, and the Java implementation classes for this model are generated.
- **org.opencoss.pam.procspec.edit** - This plugin contains a provider to display process execution models in a user interface.
- **org.opencoss.pam.procspec.editor** - This plugin provides the user interface to view instances of the model using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.
- **org.opencoss.pam.procspec.editor.dawn** - This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a file.
- **org.opencoss.pkm.refframework** - In this plugin, the process definition metamodel is defined and stored, and the Java implementation classes for this model are generated.
org.opencoss.pkm.refframework.edit - This plugin contains a provider to display process definition models in a user interface.

org.opencoss.pkm.refframework.editor - This plugin provides the user interface to view instances of the model in a tree based way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.

org.opencoss.pkm.refframework.editor.dawn - This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated model in a database instead of a file.

org.opencoss.pkm.refframework.diagram - This plugin provides the user interface to view instances of the model in a graphical way using several common viewers, and to add, remove, cut, copy and paste model objects, or modify the objects in a standard property sheet.

org.opencoss.pkm.refframework.diagram.dawn - This plugin is an extension of the previous one. It aims to communicate with the CDO Server to store the generated diagram model and also the process definition model in a database instead of a file.

2.3 Technologies used for OPENCOSS platform implementation

The software development technologies and libraries used for the implementation of both OPENCOSS platform clients and server have been described in detail in D6.6 document in the following chapters:

- “2.3 Technologies used for implementing the OPENCOSS platform server”
- “2.4 Technologies used in implementation of the clients”

2.4 Data storage implementation

The implementation of data storage facilities in OPENCOSS platform has been described in detail in D6.6 document in “2.4 CDO Server implementation” chapter.
**Figure 5:** Data storage used by OPENC OSS platform modules
3 Integration with external process tools

This chapter concerns the integration of external process management tools with the OPENCOSS platform. It consists of a set of features to allow the import of high-level process definitions (to define standard-related processes and company-wide processes), import of project-specific processes refined from the first definition, and finally import the execution of a process as it is running.

3.1 Final approach to process API implementation

The process integration API provides three functionalities (generic process definition, specific process definition, and process execution report), and two main entry points. The two entry points concern the import of the two process definitions and the import of process execution.

The main approach behind the process API definition is to allow the integration of external tools without imposing heavy constraints on the external tool side, allowing a ‘state-less’ integration plugin (e.g., an integration that does not require any storage of OPENCOSS-specific attributes on the external tool side).

This approach is important when integrating tools as ‘black box’, when the tool itself cannot be modified. This imposes however constraints on the OPENCOSS side, as we need to keep track of imported elements to be able to copy/modify/delete them.

3.2 Process definition and execution

The flow of data between the OPENCOSS platform and the external tool is always unidirectional from the external tool to the OPENCOSS platform (apart from acknowledgements).

Figure 6 shows the sequence of events to push a process definition to the OPENCOSS server. The processes that can be defined can be abstract/generic, such as company-wide processes, safety standards processes, or can be refinement of those as project processes.

The Ids used to identify the objects that need to be updated/deleted/created are the Id of the external tools. As the CCL objects themselves do not contain such field, we store them in the OPENCOSS platform in additional tables, managing the relationship between those external Ids and OPENCOSS internal Ids.

One of the key points here is that if the information that has been pushed to the OPENCOSS platform has been completed, by for example providing mapping between process elements and other OPENCOSS concepts such as pieces of evidence and safety arguments, this mapping is preserved when copying those elements into project processes, or when updating a process definition.

Process execution concerns just pushing execution information from the external tool to the OPENCOSS platform. The OPENCOSS platform then recovers the corresponding CCL process definition, and stores the execution information to make it available to the rest of the OPENCOSS modules.
3.3 Integration with Atego Process Director

The integration of Atego Process Director with OPENCOSS platform (OP) is referred hereinafter as “Atego Process Tool” or APT. It consists of the following main modules:

- A dedicated Web Site – facilitating Process & Project management and synchronisation with OP. This is the user entry point for the Atego Process Tool.
- Atego Process Director server - facilitating data management
3.3.1 Overview

The tool consists of a dedicated Web site for providing a set of features including those of Process Director and those expected by the OPENCOSS project.

The Web site gives direct access to Process Director Features and provides extended features for synchronization with the OPENCOSS platform.

Process Director Features allow the user to:
- define the “Reference model”, compliant to one or multiple standards (e.g., DO178, ISO26262, or to company standards)
- define the “Baseline model”, dedicated to a given project, possibly reusing generic processes compliant to standards
- track the execution of “Process model” of a given project against previously defined “Baseline model”

Figure 9 illustrates those features through three levels in terms of concerns around the process management: what should be done, what is planned to be done, and finally what has been done.
Figure 9: Main features in terms of process scope

At each level, extended features allow the user for:

- synchronizing the “Reference model” with the OPENCOSS reference framework,
- synchronizing the “Baseline model” with the OPENCOSS reference framework,
- updating the “Process model” with the OPENCOSS assurance project.

Figure 10 illustrates the main features in terms of use cases.

3.3.2 Functionality point of view

Atego Process Tool has been designed as a Web site which contains:

- A module for the default page for description of the project
- A module for configuring servers on both sides, Process Director and OPENCOSS
- A module for providing login page
- Modules for the master page providing the expected features previously described:
  - module for accessing directly to Process Director and its features
  - module for importing Process Definition (Reference model)
  - module for importing Project Definition (Reference Baseline)
  - module for updating Project Progress (Process model)

This design allows the user to access to the External Process tool in an easy way through the Web on computer, tablets, phones, etc.

Figure 11 shows the current implementation of the login page of Atego Process Tool. It allows a unified login for accessing to Process Director and features of the external process tool itself.
For accessing to Atego Process Tool, both servers of Process Director and of the OPENCOSS platform must be configured. For that purpose, Figure 12 shows that Login asks before choosing them.

Figure 10: Main features of APT
Figure 11: Login page* of APT (*Current Version, may change)

Figure 12: Server configuration of APT (*Current Version, may change)
Only after having configured the servers, Login can ask for credentials.

Figure 13: Current servers setting for APT (*Current Version, may change)

Figure 14: Login to APT (*Current Version, may change)
After login with personal credentials, the tool provides the expected features:

- Process & project management (using Process Director)
- Import Process into OPENCOSS (Reference model)
- Import Project into OPENCOSS (Reference Baseline)
- Update Project in OPENCOSS (Project Progress)

"Process & Project Management" page (Figure 16) allows to access directly in the current session to the tool "Process Director" for defining Process and Project.

Process Director is configured for defining Process according to the CCL metamodel in terms of Reference Activity, Artefact, Technique, etc. Figure 17 shows a Process Reference model as it has been captured from the Alstom Use Case. In this page, the process is viewed in BPMN formalism. It is typically such a process definition or reference model (What should be done) which is candidate to be imported into the OPENCOSS platform. Figure 18 shows the principle of the "Import Process Definition", which begins with the selection of the process to be imported into OPENCOSS.

Selection of the process starts the first phase which consists in the exportation of the related data from the repository of Process Director. All data are gathered and formatted into a target format according to the CCL metamodel. This phase succeeds only if the process may be fully translated into the target format, and for that checks are made and the result is given in a report.
In case of success and confirmation by the user, the second phase consists in the data importation into the REST formalism expected by the OPENCOSS platform. Finally, a report is given for confirming the success of this phase.

Exportation fails if exported data cannot be translated in the target format. On Process Director side, each element of processes must be defined according to given rules. These rules consist in declaring specific properties for mapping Process Definition and Project Definition to the CCL metamodel (e.i. RefActivity, Artefact, Technique, etc.). If those rules are not fully respected, import Process Definition is not possible and user must fix the error with Process Director.
Figure 18: “Import Process Definition” page* of APT (*Current Version, may change)

Figure 19: Import Process Definition Checks page* of APT (*Current Version, may change)

Figure 20 shows a process baseline model based on the process reference model shown in Figure 17 (Alstom Use case).
Figure 20: “Process Baseline model” page of APT

Figure 21 shows the page “Import Project Definition” or Baseline Model (What is planned to be done). The principle is exactly the same and it is not detailed here after. Export can’t fail since the export is computed from the process reference which has been already submitted to checks.

Figure 21: “Import Project Definition” page* of APT (*Current Version, may change)

Regarding the update of Process Progress or Process Model (What is currently done), the feature is not yet implemented. It would consist in updating the progress of each activity of a given project on OPENCOSS side according to the progress on Process Director side.
3.3.3 Technical overview

This chapter presents the software development technologies that have been used for implementing Atego Process Tool for OPENCOSS.

3.3.3.1 Development tool

The development tool used for the development is Visual Studio Express 2013 for Web (http://www.asp.net/vwd). This free development tool provides a subset of the functionality available in Visual Studio Professional 2012 and is specific to writing applications targeting such a Web platform. Websites created with Visual Studio Express 2013 should be compatible with Professional edition.

Visual Studio Express 2013 for Web has been used for implementing the prototype referred to as Atego Process Tool. The goal is to demonstrate the feasibility of the integration of Process Director with the OPENCOSS tool platform. C# is the programming language used.

3.3.4 Architecture of Atego Process Tool implementation

The current implementation of Atego Process Tool is a prototype, therefore its architecture and code may be subject to changes. For this reason, this chapter describes the main parts of the design and implementation views that should not be subject to changes.

3.3.4.1 Context and Architecture of APT implementation

Atego Process Tool (APT) for OPENCOSS aims to be a Web application, which means it may be used everywhere with an active Internet connection, and independently of the user devices (e.g. any web enabled devices).

Atego Process Director™ (referred to here as Process Director) is itself a Web application, and may be regarded as a part of APT because it is possible to access directly to its features from APT.

APT integration with OPENCOSS Tool Platform aims to unify features provided by Process Director and those expected from OPENCOSS point of view.

Figure 22 gives an overview of this integration in the context of the OPENCOSS prototype. At this level, the architectures of both tools Process Director and OPENCOSS platform are similar. Each one provides an API for communication between clients and servers. Process Director provides Web service based on SOAP protocol, while OPENCOSS platform provides Web service based on REST. This communication link allows APT to migrate data from Process Director to OPENCOSS server. This migration includes a translation of data to the expected format (JSON) on OPENCOSS side. APT implements ASPX pages (.NET Web forms) for defining the user interface, these pages are processed on the Web server, and the resulting HTML is sent to the user’s Web browser.

3.3.4.2 Application Interface with Process Director


The full URL of the web service depends on the Process Director Server which is used and where it is deployed.
3.3.4.3 Application Interface with OPENCOSS tool platform

OPENCOSS Tool provides a REST/JSON application interface as it is described in Chapter 3 Integration with external process tools. REST style consists in a uniform interface between applications, implementations are decoupled from the services they provide.


Figure 22: Context and Architecture of APT implementation (*Current Version, may change)
4 Project compliance estimation with Compliance report

4.1 Goal of the report

Compliance report provides extensive functionality which helps OPENCOSS platform users to assess the current compliance of their project to the selected safety standard (i.e., baseline).

The functionality is intended to be used by:

- Project team members, for example developers, when the project is in progress, in order to have up-to-date insights into which of the baseline framework items are already satisfied and to what extent.
- Project safety manager in order to monitor the project general compliance, observe the compliance details and add, assign, or un-assign specific evidence resources to/from the given requirement of the safety standard which is followed by the project.
- Independent safety assessor, when the project draws to an end, in order to browse the assigned safety evidence, evaluate it and independently assess the actual project compliance to the specific safety standard.

Two modes of the report can be distinguished:

- An interactive mode, where user can actively browse the report, select the specific baseline items, view their properties, their compliance mapping, and the associated evidence, and add or remove the evidence resources mapped to the specific baseline element.
- A printer friendly report - which is a textual output presenting all the information of the current compliance of the selected project.

The compliance report can be accessed via the following OPENCOSS web server menu item.

![Figure 23 Menu item directing to “Compliance report”](image)
4.2 Viewing compliance data on the report

The compliance report allow users to see the overall compliance of the selected project to the specific safety standard.

When a specific OPENCOSS assurance project is selected in the top panel, its defined baselines are presented in the middle panel select box.

![Baseline Frameworks combo box for the specific project](image)

Figure 24: Baseline Frameworks combo box for the specific project

The report data section is divided into 4 panels.

![4 panels of "Compliance report"](image)

Figure 25  4 panels of "Compliance report"

The “Project Compliance” table, which is placed in the left, presents base artefacts and base activities of the selected safety standard. The most important column is the “Compliance Status”
one, which presents the overall compliance status of a project to the specific safety standard item. The column can be sorted by value, thus allowing user to assess the project compliance at one glance.

In case base activities or base artefacts are defined to have a parent-child hierarchy, this relation is presented accordingly in a tree structure of the table.

Note:
“IA Status” column presents the current status of specific baseline element from Impact Analysis point of view. This functionality has been described in a separate chapter: Impact Analysis result presentation on OPENC OSS server reports.

When a specific baseline element item (i.e. table row) is selected, its description and properties, as defined in OPENC OSS storage, are presented in the bottom-left panel.

![Figure 26 Description of the selected baseline element presented at the bottom panel](image)

Upon the selection of the specific safety standard item in the “Project Compliance” table on the left of the screen, the compliance mapping details are presented in the “Base Asset Compliance Details” panel at the right side of the page.
The extensive compliance information is presented, including:

- **Compliance justification** explanation (as specified in OPENCOSS client editor or on this report).
- For the specific justification: the associated artefact or activity.
- For the specific artefact, its associated evidence resource files. These resource files are committed to the appropriate SVN repository. Users can press the [Download] link next to each resource tree node in order to download the specific file from the SVN and view it.
The above tree can be expanded or collapsed quickly to the desired level by pressing buttons above it, allowing tailoring the presented details to the level needed by a user at a given moment. When any of the above tree levels is selected (justification, artefact, or resource), its description and properties are presented in the right-bottom panel of the report.

![Figure 29: Specific evidence details description presented at the bottom](image)

### 4.3 Adding evidence and compliance data

Additionally to browsing the project evidence pieces, the report allows users to **add, modify and remove evidence** resources and define a **compliance mapping**.

“**Base Asset Compliance Details**” panel on the right-hand side of the report contains an **Upload** panel which allows users to add a specific file resource (containing the evidence), specify the associated artefact and define compliance justification text and its type.

After user presses the upload button or drag and drops the file resource to the panel, the following “**New Resource Definition**” dialog appears:
User can enter the desired **compliance justification** in the text area, change the names to be created (default names are suggested) and define the **compliance mapping type**.

Additionally it is possible to specify a **SVN URL** location where the evidence file would be committed. User has a possibility to add new location or select already defined one from the select-box.

After pressing **Assign** button, the following actions are performed by the OPENCOSS platform:

- The resource file gets **committed to** the given **SVN** repository so that it is securely stored and can be retrieved on demand.
- A **resource** CCL object (associated with the above file) is created with the specific name.
- An **artefact** CCL object (associated with the specific resource) gets created.
- A **compliance justification**, which maps the artefact to the selected baseline framework items, gets created in OPENCOSS storage.

Additionally, there are [**Modify**] and [**Unassign**] buttons, which allow user to update or revoke the evidence file and compliance mapping created above.
4.4 Generation of summary textual report

The interactive mode presented in the preceding chapters is very comfortable for users to browse and filter data, and view their details. Upon each user selection, appropriate details are presented. However, there is often a need to generate an overall report, containing all the information visualized in one place. This can be easily done using Export to MS Word button.

Upon pressing it, a default an docx template report gets filled with the all the Compliance report data presented for the specific safety project.

Note for OPENCOSS administrators:

The template docx used for textual report generation can be changed on OPENCOSS server side in order to adjust it the given company standards.

The textual report can then be printed to pdf or on paper, signed digitally or manually, and stored for future reference.
Compliance Summary Report

Date: 2016-12-12 15:33
Project name: ePark

Project Compliance Validation Summary

[Comments to be filled by the responsible person - Safety Manager or Safety Assessor]

This document contains summary of all safety evidence pieces for compliance of "ePark" project to the safety standard requirement - project benchmark "ISO 26262".

Hand Signatures
Safety Manager: .................................................................
Project Manager: ..............................................................
Build Manager: .................................................................

Figure 33 First page of the generated textual report
5 Metrics implementation

As already mentioned in D7.4, there are two orientations for OPENCOSS work towards measures: identifying the needs for information for the certification process itself (making use of data possibly not hosted in OPENCOSS), and identifying certification process data potentially of interest for other needs. However, it must be noted that all this data is not necessarily stored in OPENCOSS.

In this section, we introduce the metrics to monitor safety assurance project. All the metrics are presented with a link to the corresponding metamodel in CCL and the location of their implementation in the user interface. The metrics are created using a top-down (GQM) and bottom-up (according to our CCL) mixed approach. The detail of our approach is shown in D7.4.

5.1 Metrics selected for implementation

The selected metrics for implementation are the following, organized by Metamodels and also by charts and reports where are implemented:

**Metrics derived from Reference Framework Metamodel - Equivalence Map Report**

- **Equivalence Map Report**
  - Number of Full Maps for Requirements between two Reference Frameworks.
  - Number of Partial Maps for Requirements between two Reference Frameworks.
  - Number of No Maps for Requirements between two Reference Frameworks.
  - Number of Full Maps for Artefacts between two Reference Frameworks.
  - Number of Partial Maps for Artefacts between two Reference Frameworks.
  - Number of No Maps for Artefacts between two Reference Frameworks.
  - Number of Full Maps for Roles between two Reference Frameworks.
  - Number of Partial Maps for Roles between two Reference Frameworks.
  - Number of No Maps for Roles between two Reference Frameworks.
  - Number of Full Maps for Techniques between two Reference Frameworks.
  - Number of Partial Maps for Techniques between two Reference Frameworks.
  - Number of No Maps for Techniques between two Reference Frameworks.
  - Number of Full Maps for Activities between two Reference Frameworks.
  - Number of Partial Maps for Activities between two Reference Frameworks.
  - Number of No Maps for Activities between two Reference Frameworks.

**Metrics derived from Mapping Metamodel - Metrics Estimation Report**

- **Mapping Metrics**
  - Number of Compliance Full Maps for Requirements.
  - Number of Compliance Partial Maps for Requirements.
  - Number of Compliance No Maps for Requirements.
  - Number of Compliance Full Maps for Artefacts.
  - Number of Compliance Partial Maps for Artefacts.
  - Number of Compliance No Maps for Artefacts.
  - Number of Compliance Full Maps for Roles.
  - Number of Compliance Partial Maps for Roles.
  - Number of Compliance No Maps for Roles.
Implementation of the evidence management service infrastructure

- Number of Compliance Full Maps for Techniques.
- Number of Compliance Partial Maps for Techniques.
- Number of Compliance No Maps for Techniques.
- Number of Compliance Full Maps for Activities.
- Number of Compliance Partial Maps for Activities.
- Number of Compliance No Maps for Activities.

**Metrics derived from AssuranceAsset Metamodel - Metrics Estimation Report**

- **Assurance Asset Metrics Chart I**
  - Number of Creation Artefact Events for Evidence per week.
  - Number of Modification Artefact Events for Evidence per week.
  - Number of Evaluation Artefact Events for Evidence per week.
  - Number of Approval Artefact Events for Evidence per week.
  - Number of Revocation Artefact Events for Evidence per week.

- **Assurance Asset Metrics Chart II**
  - Total Number of Artefact Events for Evidence per day.

- **Assurance Asset Metrics Chart III**
  - Number of Creation Activity Events for Process per week.
  - Number of Modification Activity Events for Process per week.
  - Number of Evaluation Activity Events for Process per week.
  - Number of Approval Activity Events for Process per week.
  - Number of Revocation Activity Events for Process per week.

- **Assurance Asset Metrics Chart IV**
  - Total Number of Activity Events for Process per day.

**Metrics derived from Baseline Metamodel - Metrics Estimation Report**

- **Baseline Metrics Chart I**
  - Number of requirements per each applicability and criticality levels: e.g.: Number of Requirements for ASIL A ++, ASIL A +, ASIL C o, etc.

- **Baseline Metrics Chart II**
  - Number of Base Requirements.
  - Number of Base Activities.
  - Number of Base Techniques.
  - Number of Base Roles.
  - Number of Base Artefacts.

**Metrics derived from Refframework and Baseline Metamodels - Metrics Estimation Report**

- **Refframework Metrics**
  - Number of Covered Requirements of a Reference Framework to a Baseline one.
  - Number of Not Covered Requirements of a Reference Framework to a Baseline one.
  - Number of Covered Artefacts of a Reference Framework to a Baseline one.
  - Number of Not Covered Artefacts of a Reference Framework to a Baseline one.
  - Number of Covered Roles of a Reference Framework to a Baseline one.
  - Number of Not Covered Roles of a Reference Framework to a Baseline one.
  - Number of Covered Techniques of a Reference Framework to a Baseline one.
  - Number of Not Covered Techniques of a Reference Framework to a Baseline one.
  - Number of Covered Activities of a Reference Framework to a Baseline one.
  - Number of Not Covered Activities of a Reference Framework to a Baseline one.
Metrics derived from Process Metamodel - Metrics Estimation Report

- **Monitor of Process Chart I**
  - Number of Assurance Asset Activity Events per Participant.
  - Number of Assurance Asset Artefact Events per Participant.

- **Monitor of Process Chart II**
  - Number of Not Started Activities per Participant.
  - Number of Under Development Activities per Participant.
  - Number of Developed Activities per Participant.

- **Monitor of Process Chart III**
  - Number and evolution of Not Started Activities during its lifetime.
  - Number and evolution of Under Development Activities during its lifetime.
  - Number and evolution of Developed Activities during its lifetime.

Metrics derived from Argumentation Metamodel

- **Argumentation Metrics**
  - Number of All Claims.
  - Number of Public Safety Claims.
  - Number of Assumed Safety Claims.
  - Number of Un-instantiated Safety Claims in a Safety Case.
  - Number of Undeveloped Safety Claims.
  - Number of Un-instantiated and Undeveloped Safety Claims in a Safety Case.
  - Number of Claims with no Information Element Citation Associated
  - Context Information Elements.

Metrics derived from Assurance Project Metamodel

- **Time Efficiency**
  - A histogram per each activity.

- **Resource Efficiency**
  - Amount of time and number of participants per activity.

### 5.2 Metrics implementation details

In this section, we introduce our implementation specifications on the selected metrics shown in Section 5.1. The metrics are categorized according to different goals, and described with some simple examples. The goal of this section is to describe CCL entities used for each metric and calculation formulas for each metric.

*Note that, the GQM is used to develop the metrics; in the following section we only represent metrics with their original goals. For more detailed description, refer to the excel document in D7.4.*

#### 5.2.1 Metrics derived from Mapping Metamodel

<table>
<thead>
<tr>
<th>Goal</th>
<th>Maintenance of mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>This metric is used to represent the status of each type of equivalence mappings and compliance mappings.</em></td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Number of Equivalence Maps</td>
</tr>
<tr>
<td></td>
<td>- Example: 50 Reference Activities in Reference Framework X and 100 Reference Activities in Reference Framework Y, 20 of X are</td>
</tr>
</tbody>
</table>
mapped to 70 activities of Y using EQs; 10 activities of X are partially mapped to 15 of Y and the rest are not mapped. 20/50 potential reusability of X to Y; 10/50 partially reuse from X to Y; 30/50 Not reused.

- Parameters:
  - \#EquivalenceMap = Number of Equivalence Maps
- Formula:
  \[ \sum \#EquivalenceMap \]

2. Number of Compliance Maps per type (e.g., requirements, artefacts, etc.)

- Parameters:
  - \#ComplianceMap = Number of Compliance Maps per type
- Formula:
  \[ \sum \#ComplianceMap \]

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Safety Assessor, Certification Authorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Implemented</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>

5.2.2 Metrics derived from AssuranceAsset Metamodel

**Goal**

**Monitor Assurance Asset**

This metric is used for monitoring the activities related with Assurance Asset. The monitored activities include modification, creation, evaluation, revocation, etc.

**Metrics**

1. Ratio between Modification Assets and the total number of Assets

   - Parameters:
     - \#ModificationAssets = Number of Modification Assets
     - \#TotalAssets = Number of all Assets
   - Formula:
     \[ \frac{\#ModificationAssets}{\#TotalAssets} \times 100\% \]

2. Ratio between Evaluation Assets and the total number of Assets

   - Parameters:
     - \#EvaluationAssets = Number of Evaluation Assets
     - \#TotalAssets = Number of all Assets
   - Formula:
     \[ \frac{\#EvaluationAssets}{\#TotalAssets} \times 100\% \]

3. Ratio between Approval Assets and the total number of Assets

   - Parameters:
     - \#ApprovalAssets = Number of Approval Assets
     - \#TotalAssets = Number of all Assets
4. Ratio between Revocation Assets and the total number of Assets
   - **Parameters:**
     - \( \#RevocationAssets = \) Number of Revocation Assets
     - \( \#TotalAssets = \) Number of all Assets
   - **Formula:**
     \[
     \frac{\#RevocationAssets}{\#TotalAssets} \% 
     \]

5. Ratio between Creation Assets and the total number of Assets
   - **Parameters:**
     - \( \#CreationAssets = \) Number of Creation Assets
     - \( \#TotalAssets = \) Number of all Assets
   - **Formula:**
     \[
     \frac{\#CreationAssets}{\#TotalAssets} \% 
     \]
5.2.3 Metrics derived from Baseline Metamodel

**Goal**  
*Plan of certification for a given standard:*

These metrics are used to have an overall view of the Traceability of the Requirements Maintenance as well as the different kind of Requirements and its levels of criticality and applicability defined by the reference assurance framework.

**Metrics**

1. Type of defined parameters for base requirements related metrics
   - \( \#\text{BaseRequirements} \) = Number of Owned Requirements
   - \( \text{BaseActivity} \) = Base Activity Type
   - \( \text{BaseTechnique} \) = Base Technique Type
   - \( \text{BaseRole} \) = Base Role Type
   - \( \text{Base Artefact} \) = Base Artefact Type

   - **Formulas:**
     - \( \frac{\text{BaseActivity}}{\#\text{BaseRequirements}} \% \)
     - \( \frac{\text{BaseTechnique}}{\#\text{BaseRequirements}} \% \)
     - \( \frac{\text{BaseRole}}{\#\text{BaseRequirements}} \% \)
     - \( \frac{\text{Base Artefact}}{\#\text{BaseRequirements}} \% \)
2. Number of Requirements per each Base Applicability and Criticality

- Parameters:
  - $Base\text{ApplicabilityLevel} = \text{Base Applicability Level}$
  - $Base\text{CriticalityLevel} = \text{Base Criticality Level}$
  - $#Base\text{Requirements} = \text{Number of Base Requirements}$

- Formula:

$$\sum_{Base\text{Applicability} \& Base\text{Criticality}} #Base\text{Requirements}$$

---

<table>
<thead>
<tr>
<th>Status</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>
5.2.4 Metrics derived from Refframework and Baseline Metamodels

<table>
<thead>
<tr>
<th>Goal</th>
<th>Coverage of Assurance Project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The metric of the coverage between the Baseline Framework and a Reference Framework is to visually represent the relationships among them.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Per each metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• Baseline Metric</td>
</tr>
<tr>
<td></td>
<td>• Refframework Metric</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>Baseline Metric</td>
</tr>
<tr>
<td></td>
<td>Refframework Metric</td>
</tr>
</tbody>
</table>

Note that, the metrics that we are using for the Refframework Metric parameter, are the same ones as the Baseline Metamodel metrics with the exception that they are not going to be displayed.

<table>
<thead>
<tr>
<th>Status</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>

5.2.5 Metrics derived from Process Metamodel

<table>
<thead>
<tr>
<th>Goal</th>
<th>Monitor of Assurance Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The purpose of these metrics is to give a specific overview of the relations between the participants and the organizations with the work they have done. As well as the time evolution of the project.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metrics</th>
<th>1. Per each participant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #Artefact = Number of Artefacts</td>
</tr>
<tr>
<td></td>
<td>• #AssuranceAssetEvent = Number of Assurance Asset Events</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>#Artefact</td>
</tr>
<tr>
<td></td>
<td>#AssuranceAssetEvent %</td>
</tr>
</tbody>
</table>
2. Number of not started Activities per Participant
   - **Parameters:**
     - \( #Activity \) = Number of Activities
     - \( start\_time \) = Start time of an Activity
     - \( actual\_time \) = Time in which the metric has been calculated.
   - **Formula:**
     \[
     \sum_{start\_time < actual\_time} \#Activity
     \]

3. Number of Activities under development per Participant
   - **Parameters:**
     - \( #Activity \) = Number of Activities
     - \( start\_time \) = Start time of an Activity
     - \( actual\_time \) = Time in which the metric has been calculated.
     - \( end\_time \) = End time of an Activity
   - **Formula:**
     \[
     \sum_{end\_time < actual\_time \land start\_time \geq actual\_time} \#Activity
     \]

4. Number of Activities already developed per Participant
   - **Parameters:**
     - \( #Activity \) = Number of Activities
     - \( end\_time \) = End time of an Activity
     - \( actual\_time \) = Time in which the metric has been calculated.
   - **Formula:**
     \[
     \sum_{end\_time < actual\_time} \#Activity
     \]
5. Lifetime of activity

Chart 3: implementation mock-up for metric 5

5.2.6 Metrics derived from Argumentation Metamodel

<table>
<thead>
<tr>
<th>Goal</th>
<th>Monitor of Argumentation Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>These metrics are used to monitor the argumentation process and to help the user to know the status of safety argumentation, e.g., how many undeveloped safety claims are left or whether all the evidences are</td>
<td></td>
</tr>
</tbody>
</table>
connected with the safety claims.

<table>
<thead>
<tr>
<th>Metrics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Number of Safety Claims</td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #SafetyClaims = Number of Safety Claims</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>$\sum #SafetyClaims$</td>
</tr>
<tr>
<td>2. Number of Uninstantiated Safety Claims in a Safety Case.</td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #Uninstantiated = Number of Uninstantied Safety Claims</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>$\sum_{\text{safetyCase}} #Uninstantiated$</td>
</tr>
<tr>
<td>3. Number of Undeveloped Safety Claims</td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #Undeveloped = Number of Undeveloped Safety Claims</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>$\sum_{\text{safetyCase}} #Undeveloped$</td>
</tr>
<tr>
<td>4. Number of Uninstantiated and Undeveloped Safety Claims in a Safety Case</td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #Uninstantiated = Number of Uninstantied Safety Claims</td>
</tr>
<tr>
<td></td>
<td>• #Undeveloped = Number of Undeveloped Safety Claims</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>$\sum_{\text{safetyCase}} #Uninstantiated + #Undeveloped$</td>
</tr>
<tr>
<td>5. Number of Claims with no Information Element Citation associated with an asserted Evidence relationship with them on their decomposition tree (claims with no evidence associated).</td>
<td>Parameters:</td>
</tr>
<tr>
<td></td>
<td>• #Claims = Number of Claims</td>
</tr>
<tr>
<td></td>
<td>• InformationElementCitation = Information Element Citation</td>
</tr>
<tr>
<td></td>
<td>• assertedEvidence = Asserted Evidence</td>
</tr>
<tr>
<td></td>
<td>Formula:</td>
</tr>
<tr>
<td></td>
<td>$\sum_{\neg\text{InformationElementCitation}} #Claims$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>
5.2.7 Other Metrics Related to Safety Assurance and Certification Process

In D7.2, there are a number of metrics created according to industrial needs. We also cover most of those metrics in our metric implementation. In this section, we give our implementation plan for the metrics which are implemented.

5.2.7.1 Time efficiency metrics

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS allows setting and monitoring of time efficiency metrics for the assurance/certification process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS allows to set time plans (calendar working days) for the assurance/certification activities, to monitor progress, set new forecast, and to evaluate actuals against plans. The preferred indicator shall be the SPI “Schedule Performance Indicator”.</td>
</tr>
<tr>
<td></td>
<td>Goal: plan and control the certification process</td>
</tr>
<tr>
<td></td>
<td>Question: to what extent we are meeting our deadlines?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Safety Assessor, Certification Authorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Implemented with modifications</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>

**Implementation Plan**

The way we approach this metric is using the Metrics derived from the Process Metamodel; using #Activities Not Started, #Activities under development and #Activities developed with respect a fixed date (i.e. the day of the calculation of the metric).

We implemented the following representation with a Gantt chart:

![Gantt Chart Demo](image)
### 5.2.7.2 Resource efficiency metrics

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS allows setting and monitoring of resource efficiency metrics for the assurance/certification process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS allows setting resources plans (man/hours) for the assurance/certification activities, to monitor progress, set new forecast, and to evaluate actuals against plans. The preferred indicator shall be the CPI “Cost Performance Indicator”. The assurance/certification process is mainly a human activity (occasionally supported by software tools). Hence man/hours do represent the most significant metrics for resource usage. Goal: plan and control the certification process Question: to what extent we are meeting our effort budget?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Safety Assessor, Certification Authorities</td>
</tr>
<tr>
<td>Status</td>
<td>Implemented with modifications</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
<tr>
<td>Implementation Plan</td>
<td>We retrieve information from the Process Metamodel in order to obtain the duration in hours per activity. Also we know the number of participants per activity. Note that, the CCL does not support the effort dedicated to the activity by each participant, it only supports the total effort of the activity. Therefore, we approach this metric in a grosser way, getting the total number of participants and the total duration, in hours, of an activity.</td>
</tr>
</tbody>
</table>

### 5.2.7.3 Support metrics estimation

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS supports the estimation of support metrics for the assurance/certification process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS supports the estimation of schedule and resources of a new assurance/certification process executed in a particular project as the result of a parametric model and/or statistical historical data. Goal: plan and monitor the certification process Question: what will be the schedules and efforts for this new certification project?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Safety Assessor, Certification Authorities</td>
</tr>
<tr>
<td>Status</td>
<td>Cancelled - It is not CCL related, therefore, there is no available data in the database to calculate the metric.</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
</tbody>
</table>
5.2.7.4 Completeness metrics estimation

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS supports the estimation of completeness (effectiveness) metrics for the assurance/certification process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS supports completeness by collection of metrics on safety problems and safety problems removal efficiency, organized by risks levels. Goal: improve the certification process Question: to what extent are we effective in finding safety related problems?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Safety Assessor, Certification Authorities</td>
</tr>
<tr>
<td>Status</td>
<td>Implemented</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
<tr>
<td>Implementation Plan</td>
<td>This metric is covered by the Baseline Metamodel and Argumentation Metamodel metrics.</td>
</tr>
</tbody>
</table>

5.2.7.5 Compliance coverage metrics estimation

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS supports the estimation of the compliance coverage metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS Platform supports progress indicators that will provide informative data about the status of the compliance to selected standard. The metrics is a ratio of completed and compliant items/requirements against overall expected items. It would be desirable that items are weighted in terms of required effort. Goal: plan and control the certification process Question: how much progress we have?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Safety Assessor, Certification Authorities</td>
</tr>
<tr>
<td>Status</td>
<td>Implemented</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
<tr>
<td>Implementation Plan</td>
<td>This metric is covered by the Compliance Map Metamodel metrics.</td>
</tr>
</tbody>
</table>
5.2.7.6 Claims coverage metric estimation

<table>
<thead>
<tr>
<th>Description</th>
<th>OPENCOSS supports the estimation of the claims coverage metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>OPENCOSS Platform supports the collection of data about claims demonstrated and ratio over all claims.</td>
</tr>
<tr>
<td></td>
<td>Goal: plan and control the certification process</td>
</tr>
<tr>
<td></td>
<td>Question: how much progress we have?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Safety Assessor, Certification Authorities</td>
</tr>
<tr>
<td>Status</td>
<td>Implemented</td>
</tr>
<tr>
<td>Priority</td>
<td>M</td>
</tr>
<tr>
<td>Implementation Plan</td>
<td>This metric is covered by the Argumentation Metamodel metrics.</td>
</tr>
</tbody>
</table>

5.3 Metrics Estimation Report

This chapter presents the implemented metric reports, their functionality and layout. The Metrics Estimation Report can be accessed via the following OPENCOSS web server menu item:

![Menu item directing to "Metrics Estimation report"](image)

Figure 32: Menu item directing to "Metrics Estimation report"

When a specific OPENCOSS safety project is selected in the top panel, its defined baselines are presented in the middle panel select box, as shown in Figure 24.

The report data is divided into two panels. The first one, on the left, is a static menu panel in which the user can select a type of metrics to analyse.
Once a specific metrics is selected on the Metrics menu, the metrics menu details are presented in the right part of the report with a description of the main goal and all the different types of charts related to that metric.

Also, the user has the option to the selected metric to a Word Document with more detailed information.

As a small example, the following figure:
5.4 Equivalence Map Report

This chapter presents the implemented equivalence map report metrics, their functionality and layout. The Equivalence Map Report can be accessed via the following OPENCQSS web server menu item as showed beforehand.

In this case the equivalence metrics are between two Reference frameworks, not specific OPENCQSS safety project needs to be selected in the top panel. The only possible configurations are between reference frameworks as shown in Figure:

After the selection, the metrics of the equivalence maps and a detailed description are showed on the screen. There is also a possibility to export this information to a document.
5.5 Source code description

This chapter describes implementation source code details. As aforementioned, this source code is also hosted at the SVN OPENCOSS repository. For now, all the metrics are implemented in the Server source code in the package “org.opencoss.webapp.reports” - refer to Figure 3.

Part of this package contains the implementation of the Metrics server web pages. The functionality has been developed using Vaadin, which is a Java framework for building modern web applications (http://www.vaadin.com), and also with combination of JFreeChart, which is an open-source framework for the programming language Java, which allows the creation of a wide variety of both interactive and non-interactive charts (http://www.jfree.org/jfreechart).

From WP7 functionality perspective, the org.opencoss.webapp.reports package contains the implementation of Metrics Estimation report and Equivalence Map report. Most of its source code has been described in Chapter 2.2.1 but not the Metric’s View component according to Model-View-Controller (MVC) design pattern.

MetricsEstimationReport plays a role of a controller and is a hub for both data presentation and its retrieval from data storage. The getMainComponent() method assembles a view - GUI of the report - which consists of two panels. The details about metrics estimation GUI and its functionality have been described in the above chapter.
The GUI left-handed-side Menu represents the different kind of metrics that have been implemented in the project. Each entry is a combination of different types of charts and tables in order to represent as much information as possible about the project selected in the Project's Combo Box and the Baseline Framework selected.

As above-mentioned, the model component of MVC pattern is materialized by **Data-Access-Objects (DAO) design pattern** classes implemented in `org.opencoss.webapp.reports.dao` package. The Metrics report uses those classes in order to prepare SQL queries and use CDO connectivity functions to retrieve data from OPENC OSS storage and later on, shape that raw data to represent it in a graphical way.

Finally, the following figure represents a global overview of Opencoss Metrics Implementation.

**Figure 37:** Metrics implementation layers
6  OPENCOSS platform tool user manual

The manual describing how to use all the implemented functionalities of the OPENCOSS platform can be found in the OPENCOSS SVN repository at:

7 OPENCOSS platform tool developer manual

Guides for developers, including step-by-step instructions of how to set up development environments can be found at the following place, for OPENCOSS tool client and server respectively:

8 Conclusion

This document has summarized and presented the details of the software development performed to implement process management, metrics evaluation, and compliance estimation functionality in OPENCOSS platform tools.

Major functional areas defined in D7.2 document has been implemented in the software prototype. It has been decided that the functionality of process definition and execution would be facilitated by the external process tool. In order to achieve this integration, a special bridge has been implemented to connect to Atego Process Director. The functionality of compliance estimation and metrics have been implemented as reports on OPENCOSS web server.

This deliverable document has shown or has referenced to the actual descriptions already done in D6.6 document: the exact implementation architecture, software technologies used and their configurations. It has described the most important pieces of the developed code and referred to specific source code in the repository. Additionally, a user manual and software development guides have been pointed.